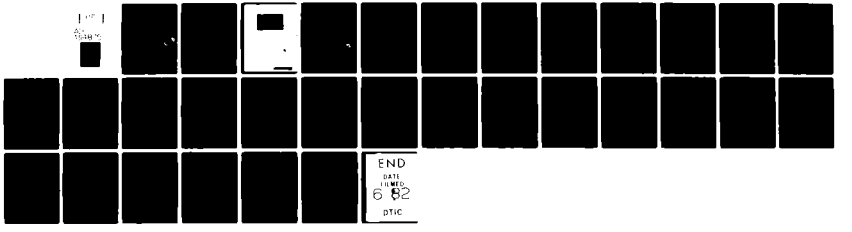
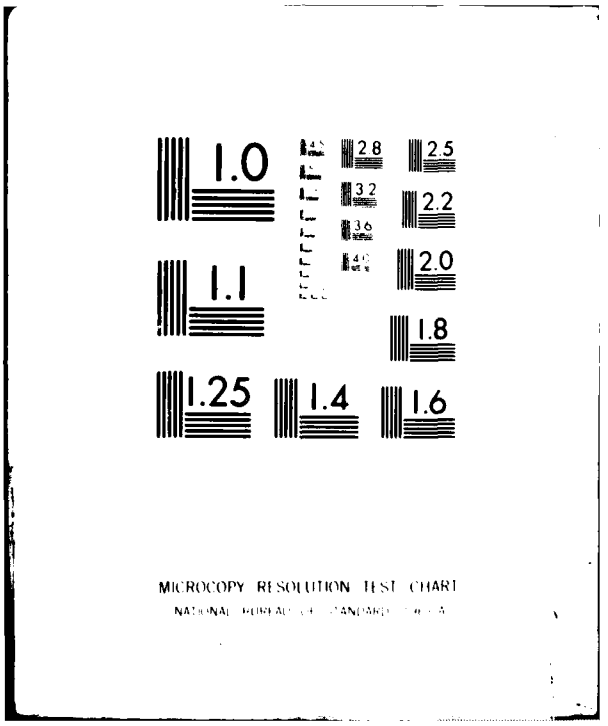
AD-A114 875 HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB F/G 12/1
THE EXPECTED TIME COMPLEXITY OF PARALLEL GRAPH AND DIGRAPH ALGO--ETC(U)
APR 82 J H REIF, P SPIRAKIS N00014-80-C-0674
UNCLASSIFIED TR-11-82 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
MICROCOPY RESOLUTION TEST CHART

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	AD-A114875	

4. TITLE (and Subtitle)

The Expected Time Complexity of Parallel Graph and Digraph Algorithms

5. TYPE OF REPORT & PERIOD COVERED

Technical Report

6. PERFORMING ORG. REPORT NUMBER

TR-11-82

7. AUTHOR(s)

John H. Reif
Paul Spirakis

8. CONTRACT OR GRANT NUMBER(s)

N00014-80-C-0674

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Harvard University
Cambridge, MA 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

12. REPORT DATE

April, 1982

13. NUMBER OF PAGES

25

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

same as above

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

unlimited

This document has been approved for public release and sale; its distribution is unlimited.

DTIC ELECTE MAY 26 1982 S D H

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

random graph, random digraph, parallel algorithms, connectivity, transitive closure, minimum cost paths, biconnected components, minimum spanning trees, graph isomorphism, expected time, parallel algorithms, parallel RAM.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A114875

DTIC FILE COPY

20.

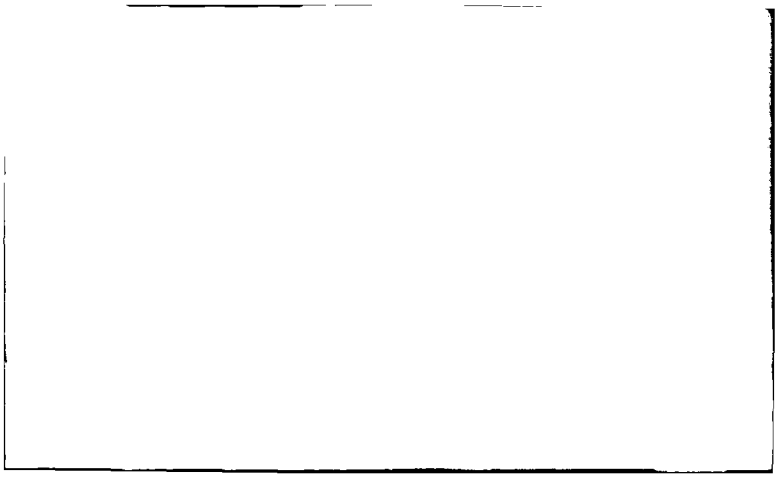
This paper determines upper bounds on the expected time complexity for a variety of known parallel algorithms for graph problems.

For connectivity of both undirected and directed graphs, transitive closure and all pairs minimum cost paths, we prove the expected time is $O(\text{loglog } n)$ for a parallel RAM model (RP-RAM) which allows random resolution of write conflicts, and expected time $O(\log n \text{ loglog } n)$ for the P-RAM of [Wyllie, 79], which allows no write conflicts.

We show that the expected parallel time for biconnected components and minimum spanning trees is $O((\text{loglog } n)^2)$ for the RP-RAM and $O(\log n \cdot (\text{loglog } n)^2)$ for the P-RAM.

Also we show that the problem of random graph isomorphism has expected parallel time $O(\text{loglog } n)$ and $O(\log n)$ for the above parallel models, respectively.

Our results also improve known upper bounds on the expected space required for sequential graph algorithms. For example, we shown ~~in section~~ 7 that the problems of finding strong components, transitive closure and minimum cost paths have expected sequential space $O(\log \cdot \text{loglog } n)$ with $n^{O(1)}$ time on a Turing Machine given random graphs as inputs.



THE EXPECTED TIME COMPLEXITY OF PARALLEL GRAPH AND DIGRAPH ALGORITHMS

John H. Reif

Paul Spirakis

TR-11-82

April, 1982

THE EXPECTED TIME COMPLEXITY OF PARALLEL GRAPH AND DIGRAPH ALGORITHMS

by

John Reif* and Paul Spirakis
Aiken Computation Laboratory
Division of Applied Sciences
Harvard University, Cambridge, Massachusetts

SUMMARY

This paper determines upper bounds on the expected time complexity for a variety of known parallel algorithms for graph problems.

For connectivity of both undirected and directed graphs, transitive closure and all pairs minimum cost paths, we prove the expected time is $O(\log\log n)$ for a parallel RAM model (RP-RAM) which allows random resolution of write conflicts, and expected time $O(\log n \log\log n)$ for the P-RAM of [Wyllie, 79], which allows no write conflicts.

We show that the expected parallel time for biconnected components and minimum spanning trees is $O((\log\log n)^2)$ for the RP-RAM and $O(\log n \cdot (\log\log n)^2)$ for the P-RAM.

Also we show that the problem of random graph isomorphism has expected parallel time $O(\log\log n)$ and $O(\log n)$ for the above parallel models, respectively.

Our results also improve known upper bounds on the expected space required for sequential graph algorithms. For example, we shown in Section 7 that the problems of finding strong components, transitive closure and minimum cost paths have expected sequential space $O(\log \cdot \log\log n)$ with $n^{O(1)}$ time on a Turing Machine given random graphs as inputs.

---

*This work was supported by the National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0674.

RANDOM INPUT

	PROBLEM	P-RAM	RP-RAM
UNDIRECTED	Connectivity	$\bar{\tau}=O(\log n \log\log n)$ $\bar{\pi}=O(n^2/\log n \cdot \log\log n)$	$\bar{\tau}=O(\log\log n)$ $\bar{\pi}=O(n+pn)$
	Biconnected Components	$\bar{\tau}=O(\log n (\log\log n)^2)$ $\bar{\pi}=O(n^3/\log n)$	$\bar{\tau}=O((\log\log n)^2)$ $\bar{\pi}=O(n^3/\log n)$
	Min Cost Spanning Trees	$\bar{\tau}=O(\log n (\log\log n)^2)$ $\bar{\pi}=O(n^2/\log\log n)$	$\bar{\tau}=O((\log\log n)^2)$ $\bar{\pi}=O(n^2/\log\log n)$
	All Pairs Shortest Paths	$\bar{\tau}=O(\log n (\log\log n)^2)$ $\bar{\pi}=O(n^3/\log n)$	$\bar{\tau}=O((\log\log n)^2)$ $\bar{\pi}=O(n^3/\log n)$
DIRECTED	Transitive Closure	$\bar{\tau}=O(\log n \log\log n)$ $\bar{\pi}=O(n^3/\log n)$	$\bar{\tau}=O(\log\log n)$ $\bar{\pi}=O(n^3/\log n)$
	Strongly Connected Components	$\bar{\tau}=O(\log n \log\log n)$ $\bar{\pi}=O(n^2/\log n)$	$\bar{\tau}=O(\log\log n)$ $\bar{\pi}=O(n^3/\log n)$
	Isomorphism	$\bar{\tau}=O(\log n)$ $\bar{\pi}=O(n^2/\log n)$	$\bar{\tau}=O(\log\log n)$ $\bar{\pi}=O(n^2/\log\log n)$

Table 1. Results of this Paper for Random Graphs

$\bar{\pi}$ = expected number of processors for random input.

$\bar{\tau}$ = expected parallel time for random input.

WORST CASE INPUT

	PROBLEM	P-RAM	SP-RAM	PP-RAM
UNDIRECTED	Connectivity	$\tau=O(\log n \log d)$ $\pi=O(n^3/\log n)$ [Ja'Ja',78] $\tau=O(\log^2 n)$ $n=O(n^2/\log^2 n)$ [Chin et al. 81]	$\tau=O(\log n)$ $\pi=O(n+m)$ [Shiloah, Vishkin,81]	$\hat{\tau}=O(\log n)$ $\pi=O(n^3 \log n)$ [Reif,81]
	Biconnected Components	$\tau=O(\log n \log d \log k)$ $\pi=O(n^3/\log n)$ [Ja'Ja',81]	$\tau=O(\log d \log n)$ $\pi=O(n^3/\log n)$ [Ja'Ja',78]	$\hat{\tau}=O(\log n)$ $\pi=O(n^3 \log n)$ [Reif,82]
	Min Cost Spanning Trees	$\tau=O(\log^2 n)$ $n=O(n^2)$ [Savage,Ja'Ja',81]	$\tau=O(\log^2 n)$ $\pi=O(n^3)$	$\hat{\tau}=O(\log n)$ $\pi=n^{O(1)}$ [Reif,82]
	All Pairs Shortest Paths	$\tau=O(\log^2 n)$ $n=O(n^3)$ [Dekel et al.,81]	$\tau=O(\log n \cdot \log\log n)$ $\pi=\theta(n^3)$	-
DIRECTED	Transitive Closure	$\tau=O(\log n \log d)$ $\pi=O(n^3/\log n)$ [Ja'Ja",78]	$\tau=O(\log d)$ $\pi=O(n^3/\log n)$	-
	Strongly Connected Components	$\tau=O(\log n \log d)$ $\pi=O(n^3/\log n)$ [Ja'Ja',78]	$\tau=O(\log d)$ $\pi=O(n^3/\log n)$ [Ja'Ja',78]	-
	Isomorphism	-	-	-

Table 2. Previously Known Results for Worst Case Input

π is the number of processors on worst case.
τ is parallel time for worst case input.
$\hat{\tau}$ is expected parallel time for worst case input.

1. INTRODUCTION

1.1 The Problems

Considerable work has been done on sequential graph algorithms which are fast on the average given random input graphs. This includes the work of [Angluin, Valiant, 79], [Karp, 76], [Karp, Sipser, 81], [Schnorr, 78], [Karp, Tarjan, 80], [Reif, Spirakis, 80], [Spirakis, 81]. Almost no previous work examined the average performance of parallel graph and digraph algorithms.

We analyze here the average performance of parallel algorithms for connectivity, biconnected components, strong connectivity and transitive closure, minimum cost spanning tree, minimum cost all pair shortest paths and graph isomorphism.

1.2 The Parallel Machine Models

We consider here some fundamental models of parallel computation, all of which assume the presence of an unlimited number of processors. In the first model, *P-RAM* of [Wyllie, 79], each processor is capable of performing arithmetic, boolean and certain read and write operations. The processors have access to a common main memory. In the P-RAM, different processors can read the same memory location at the same time. They may store information at different memory locations simultaneously, but no two processors can attempt to change the contents of the same memory cell at the same time. [Reif, 81] presented a probabilistic P-RAM model, PP-RAM, where processors are capable of doing independent probabilistic choices on a fixed input (but, again, simultaneous writing at the same location is not allowed).

In the second model, the *SP-RAM*, simultaneous access to the same memory location is allowed for both read and write operations. In the last case

exactly one processor succeeds but we make no assumption of which one succeeds (see [Shiloah, Vishkin, 80]).

In the third model, the *RP-RAM*, again we allow simultaneous access to the same memory location for both read and write operations. In the last case, if k processors attempt to write at the same time, then exactly one succeeds. The probability that each particular one succeeds is $1/k$. (One could imagine that processors are ordered in some sequential order (all sequences being equiprobable) and then each subsequent write overwrites the effect of the previous one and that this sequential execution occurs in a very fast way, say one step.)

It is clear that any parallel algorithm for the SP-RAM requires the same expected time on the RP-RAM. It is also clear that any parallel algorithm in the RP-RAM of time T implies a parallel algorithm on the P-RAM model, of time at most $O(T.\log m)$ where m is the maximum number of processors which could possibly compete for the same memory location (one could simulate the nondeterministic choice by a tree of pairwise selections to find a unique winner).

1.3 New Results in the Theory of Random Graphs

The input to most of the algorithms here is assumed to be a random graph of the model $G_{n,p}$ as defined in [Erdös, Rényi, 60], or a random digraph $D_{n,p}$ as defined in [Spirakis, Reif, 81]. For the minimum spanning tree and shortest paths algorithms the graph is assumed to be a random graph $G_{n,p}$ with edge weights selected from a continuous distribution, independently of each other. In the $G_{n,p}$ model the probability of existence of an edge p is $\geq c/n$ where c is a constant >1. The key to the analysis of the algorithms of this paper is some new results on random graphs (given in the Appendix).

(1) The *depth of a random graph* of the model $G_{n,p}$ with $p \geq c/n$ (or $G_{n,N}$ with $N \geq cn$) is $O(\log n)$ (and decreases as the graph density becomes larger), with high probability.

(2) The *number of connected components of a random graph* (or digraph for strong connectivity) of the model $G_{n,p}$ with $p \geq c/n$ (where $c > 2$) is $O(\log n)$ with probability $\geq 1 - n^{-2}$ and this number decreases as p grows. (This holds also for k-blocks, $k \geq 1$ for any constant k.)

(3) A random graph of density $p \geq c/n$ where $c \geq 2$ has a large component (of the appropriate type) of size $\geq n - \log n$ with probability $\geq 1 - n^{-2}$. A previous result of [Karp, Tarjan, 80] showed only that there is a large component of size at least εn, $\varepsilon > 0$, with high probability.

1.4 Expected Time Results for Known Parallel Graph Algorithms

Using the above facts and some non-trivial average case analysis, we show that one can find the connected components of a random graph $G_{n,p}$ and the strong components and transitive closure of a random digraph $D_{n,p}$ and also the all-pairs shortest paths of a weighted random graph in average parallel time $O(\text{loglog } n)$ in the RP-RAM and $O(\log n \cdot \text{loglog } n)$ in the P-RAM. The average time for biconnected components and minimum spanning tree is $O((\text{loglog } n)^2)$ for RP-RAM and $O(\log n \cdot (\text{loglog } n)^2)$ for the P-RAM. The average time for graph isomorphism is $O(\text{loglog } n)$ in RP-RAM and $O(\log n)$ in P-RAM. The expected number of processors for undirected connectivity is $O(\frac{n^2}{\log n \cdot (\text{loglog } n)})$. For directed connectivity and transitive closure $O(n^3/\log n)$ processors are expected. For biconnected components we require $O(n^3/\log n)$ processors. For minimum spanning tree and all pairs shortest paths, we require $O(n^2/\log n)$ and $O(n^3/\log n)$ processors. For graph isomorphism we need $O(n^2/\log n)$ processors. Table 1 gives our result.

We remark there is generally a considerable drop in time complexity when one concentrates in the average case of the above problems rather than worst case input. For sake of comparison we list here previous results and also provide a table (Table 2). The best known deterministic algorithm for graph connectivity is $O(\log^2 n)$ time in P-RAM with $O(n^2/\log^2 n)$ processors [Hirschberg, Chandra, Sarwate, 79] and $O(\log n)$ time in SP-RAM with $O(n+m)$ processors [Shiloah, Vishkin, 80]. [JaJa, 78] has an $O(\log n \log d)$ time and $O(n^3/\log n)$ processors algorithm for the P-RAM where d is the depth of the graph. For the biconnected components, [Savage, JaJa,81] give an $O(\log^2 n)$ time algorithm in deterministic P-RAM with $O(n^3/\log n)$ processors and also a $O(\log n \log d \log k)$ time algorithm with $O(mn+n^2 \log n)$ processors where k is the number of components and m = edges of the graph. For the minimum spanning tree [.Savage,Jaja, 81] given an $O(\log^2 n)$ time algorithm with $O(n^2)$ processors. For all the above undirected graph problems, [Reif, 82] gives an $O(\log n)$ probabilistic algorithm for the PP-RAM. This time is slightly lower than the $O(\log n \log\log n)$ expected time bound of this paper for solving undirected graph problems on P-RAM, but [Reif, 82] requires an $O(n^3 \log n)$ number of processors whereas our paper requests on the average considerably less processors, on the P-RAM model. For transitive closure and strong components the best result ([JaJa, 78]) for the P-RAM is $O(\log n \log d)$ time with $O(n^3/\log n)$ processors. For all pairs shortest paths in digraphs, [Dekel, Nassimi, Sahni, 81] give an $O(\log^2 n)$ algorithm with $O(n^3)$ processors. No parallel algorithm for isomorphism was known up to now. Note also that the results of [Reif, 82] are not applicable to digraph problems.

2. EXPECTED TIME COMPLEXITY FOR PARALLEL TRANSITIVE CLOSURE AND STRONG CONNECTIVITY

The following well known algorithm for computing the transitive closure of an $n \times n$ boolean matrix A was shown in [Dekel, Nassimi, Sahni, 81] to be of $O(\log^2 n)$ worst case time by using $O(n^3)$ processors, in the P-RAM model:

Algorithm TC(A)

$B \leftarrow A + I$

L: $B' \leftarrow B$

$B \leftarrow B \cdot B$

if $B' \neq B$ then go to L

[JaJa, 78] did an efficient implementation of this algorithm showing that it could be done in $O(\log n \log d)$ time on the P-RAM model, and by using $O(n^3/\log n)$ processors, where d = depth of G. Note that we can drop the number of processors to $O(n^{\log 7}/\log n)$ by applying Schonage-Strassen algorithm for matrix multiplication. It is easy to observe that the basic set operation of [JaJa, 78] (i.e., the union of $O(n)$ sets of $O(n)$ integers each, from 1 to n) can be done in $O(1)$ time in the SP-RAM, implying an $O(\log d)$ algorithm for transitive closure. Using our results about the average depth of $D_{n,p}$ (see Appendix) we immediately have:

THEOREM 2.1. *The average parallel time for transitive closure of directed graphs is* $O(\text{loglog } n)$ *for the SP-RAM (and thus also the RP-RAM) parallel computers and uses* $O(\log n \cdot \text{loglog } n)$ *for the P-RAM model using* $O(n^3/\log n)$ *processors. The probability that the parallel time is more than this is less than* $\exp(-kn^{\alpha})$ *for* $k, \alpha > 0$.

[JaJa, 78] gave also an algorithm for finding the strong components of a digraph in parallel time $O(\log n \log d)$ in the P-RAM model, by using $O(n^3/\log n)$ processors. Our results about the depth of $D_{n,p}$ then imply:

THEOREM 2.2. *The average parallel time for finding the strong components of a digraph in the P-RAM model is* $O(\log n \log\log n)$, *using* $O(n^3/\log n)$ *processors. In the SP-RAM model (and thus also the RP-RAM) the average time is* $O(\log\log n)$ *using the same number of processors. The probability that the parallel time is more than this, is less than* $\exp(-kn^{\alpha})$ *for some* $k,\alpha > 0$.

Note that the same closure idea can be applied to undirected graphs to compute the connected components in $O(\log n \log d)$ time in the P-RAM model and $O(\log n \log\log n)$ average time, as [JaJa, 78] remarks. However, the number of processors used is wasteful and we give in the next section an independent derivation of our result which optimizes over the number of processors.

3. EXPECTED PARALLEL TIME FOR CONNNECTIVITY AND BICONNECTED COMPONENTS IN UNDIRECTED GRAPHS

3.1 Previous Parallel Algorithms for Undirected Connectivity

[Reif, 82] provided a probabilistic $O(\log n)$ time algorithm for the PP-RAM model, but requiring $O(n^3 \log n)$ processors. [JaJa, 78] and [Savage, JaJa, 81] presented algorithms in the P-RAM model for undirected graph connectivity, with parallel time $O(\log n \log d)$, where d is the graph depth, and with $O(n^3/\log n)$ processors. For both these algorithms the number of processors is wasteful and [Hirschberg, Chandra, Sarwate, 79] presented an algorithm which runs in the P-RAM model in parallel time $O(\log^2 n)$ and only using $n\lceil n/\log n \rceil$ processors. [Savage, JaJa, 81] remarked that this algorithm is really of time $O(\log n \cdot y)$ where $y = \min\{\log n, \frac{d}{2}\}$ and d is the graph depth.

(Note that thus our result that the expected d is O(log n) does not imply their algorithm has expected time below $O(\log n)^2$.) [Chin, Lam, Chen, 81] improved the algorithm of Hirschberg *et al.* to run in $O(\log^2 n)$ time by using only $O(n^2/\log^2 n)$ processors, again in the P-RAM model. [Shiloah, Vishkin, 81] adopted the Hirschberg *et al.* algorithm to run on an SP-RAM computer and provided an O(log n) algorithm by using O(n+m) processors, where m is the number of edges.

3.2 Graph Connectivity in O(loglog n) Average Parallel Time in the RP-RAM, Using $O(n^2/\log n \cdot \log\log n)$ Processors

We prove here:

THEOREM 3.1. *The average parallel time complexity for connected components of the random graph* $G_{n,p}$ *with* $p \geq \frac{c}{n}$, $c>1$, *is* O(loglog n) *in the RP-RAM model. The average number of processors used is* O(n+pn). *The probability that the time complexity exceeds* O(loglog n) *is* $o(\frac{1}{n})$.

We shall first analyze here the average performance of the [Shiloah, Vishkin, 81] algorithm, executed in our RP-RAM model. The algorithm uses 2n+m processors, hence it uses n(2+pn) processors on the average. An improvement over this will be shown later. During the computation each vertex v has a pointer field D(v) through which it points on another vertex or to itself. One can regard $v \rightarrow D(v)$ as a (directed) edge in an auxiliary graph, called the *pointers-graph*. This graph is a collection of sets, each of which is a rooted tree with a self-loop at the roots. As the algorithm proceeds, the number of sets decreases while each individual set increases (or disappears). This is caused by a "hooking" operation in which a tree is hooked on another tree. The trees are also subject to another transformation of collapsing towards the root. At the end, each remaining set is a connected component of the graph and looks like a rooted star in the pointers graph.

3.3 Informal Algorithm Description (for details see [Shiloah, Vishkin, 80])

The notation $D_s(i) = j$ means that vertex i points on vertex j after the s-th stage. Each stage has four steps, and each step takes constant time in the RP-RAM model:

Step 1: For all vertices i $D_s(i) \leftarrow D_{s-1}(D_{s-1}(i))$ (collapsing trees).

Step 2: Hook trees on smaller vertices of other trees: All the vertices that have pointed on a root at the end of the previous stage check whether their neighbors are pointing on smaller vertices. If one finds such a neighbor j, it tries to hook its tree on $D_s(j)$. At this point simultaneous writing at the same location is used by the RP-RAM and one succeeds (with equal probability). Let us call a tree *stagnated* in the s-th stage if it has not been changed in the first two steps of the stage. A root of such a tree is called a stagnated root.

Step 3: Hooking stganated trees: All the vertices that point on a stagnated root check whether their neighbors point on a vertex of another tree. If one finds such a vertex j then it tries to hook its tree on $D_s(j)$. One of them eventually succeeds.

Step 4: Second collapsing: $D_s(i) \leftarrow D_s(D_s(i))$.

We allocate processors to vertices and to edges. Each edge $\{i,j\}$ is viewed as two pairs $\langle i,j\rangle$, $\langle j,i\rangle$ and we allocate one processor to each pair. It is clear that just after Step 1 or after Step 4 the trees are rooted stars. Stagnated trees are also rooted stars at the beginning of Step 3.

Let us call each rooted tree a "supervertex." It has been remakred in the literature that the number of supervertices per connected component reduces by at least two in each stage, thus leading to a total of $O(\log n)$ number of stages and an $O(\log n)$ parallel time algorithm in the SP-RAM model.

3.4 Expected Time Analysis

We assume here that the input to the algorithm is an instance G of the random graph $G_{n,p}$ with $p \geq c/n$ and $n > 1$. We shall show that only $O(\log\log n)$ stages are needed on the average. Let us assume that there are m supervertices at the end of the t-th stage, of sizes $s_1^t, \ldots, s_m^t$. Since $p \geq c/n$, by results of the Appendix of this paper, the average valence of each node is $\geq 2c$. In fact, a large fraction ($\geq \varepsilon \cdot s_i^t$ with ε close to 1) of nodes of each set i of size s_i^t will each have valence ≥ 2 with probability tending to 1 as n tends to ∞ (since $c > 1$). The edges out of each node are placed in a random way. Let us consider a particular set, of size s_i^t. This set has a total of $2cs_i^t$ edges on the average, originating out of its vertices (it has, in fact, $\geq 2\varepsilon\, s_i^t$ edges out of its vertices with high probability). The conditional probability that each of these edges hits a particular other set of size s_k^t is obviously s_k^t/n. Hence, the total probability that the set s_k^t is hit is approximately $(2cs_i^t)(s_k^t)/n$, conditioned on the assumed sizes and valences. (In fact, it is $> (2\varepsilon s_i^t)\,(s_k^t/n)$ with high probability). Hence, the average number of the sets to which the set of size s_i^t has edges is

$$\sum_{k \neq i} (2cs_i^t)\,\frac{s_k^t}{n} \;=\; 2cs_i^t\left(\frac{n - s_i^t}{n}\right) \geq 2c(s_i^t - 1) \geq cs_i^t \quad ,$$

if we inductively assume that $2 \leq s_i^{t'} \leq \sqrt{n}$ for every i and any stage before the end of the algorithm. So, we remark that, on the average, each set (i.e., supervertex) can (in the next stage) potentially merge with at least as many supervertices as its size, multiplied by c. Let us now make the inductive hypothesis that all (but a small fraction) of the sets are of the same average size (say s^t) at time t. For each set of size S then, at least cS processors propose for it to merge with some other set. A particular processor

wins with probability $q = 1/cS$. Thus, the average number of proposals a set accepts will be at least $1/2q = cS/2$. (To understand the above argument, consider an undirected graph whose vertices have degree cS. Each vertex selects one edge *out of it* with probability $1/cS$. Then, the average *indegree* of the resulting directed graph will be at least $cS/2$.) Hence, at the next t+1-th stage, the size of that set (and of all but a small fraction of the remaining sets) will be at least $S + S \cdot \frac{cS}{2} \geq \frac{c}{2} S^2$. So, we have shown that, at the next stage, the sets will have again equal average sizes (except for a small fraction) and that $S^{t+1} \geq \frac{c}{2} (S^t)^2$. Let t_0 be the number of stages the algorithm does on the average. We have $S^{t_0} = \max_i \{S_i^{t_0}\} \leq n$, implying $t_0 = O(\text{loglog } n)$, proving our claim about the average number of stages of the algorithm. Note that the above arguments can be used to prove the giant component theorem for random graphs (see [Karp, Tarjan, 80], [Reif, Spirakis,81]).

For a more detailed analysis, the following statements can be proved by induction on the number of iterations.

(1) The size of each of the sets at the end of the t-th stage, follows approximately a Poisson distribution of mean $(c/2)^t \cdot 2^{2^t}$.

(2) The sizes of all (but a small fraction equal to $1 - \beta^{2^t}$, where $\beta < 1$) sets have small differences with probability going to 1 as $n \to \infty$.

Clearly, these two statements again imply that the mean value of number of stages is $O(\text{loglog } n)$ and that the probability that $t > \text{loglog } n$ is $o(1/n)$.

3.5 An Improvement Over the Number of Processors

[Chin, Lam, Len, 81] use arrays to represent the supervertices and the selection of the minimum neighbor in Step 2 is done in P-RAM by using nk processors in total, where $k \leq n/2$.

It is shown in [Chin, Lam, Len, 81] that the minimum of n elements can be computed on a P-RAM in time T where

$$T = \lceil n/k \rceil - 1 + \log k \qquad \text{for } k \leq \frac{n}{2} \quad .$$

For the total time bound, we shall have (on the average)

$$T_{total} = \sum_{i=0}^{\text{loglog } n} \left(\frac{N_i}{k} - 1 + \log k \right)$$

where $N_i = (c/2)^i . 2^{2^i}$

$$\leq O\left(\frac{n}{k} + (\text{loglog } n)\log k\right) \quad .$$

The optimal value of the total time is $O(\log n \text{ loglog } n)$ *by using an expected* number of $nk = n^2/\log n \cdot (\text{loglog } n)$ processors. Note that we could get the same average time complexity for the P-RAM model by just simulating our algorithm for the RP-RAM model but this would require us to use an average of $n(1+pn)$ processors.

COROLLARY 4.1. *The average parallel time for connected components of the random graph* $G_{n,p}$ *with* $p \geq c/n$ *is* $O(\log n \cdot \text{loglog } n)$ *in the P-RAM model and the number of processors needed is* $O(n^2/\log n \cdot \text{loglog } n)$ *on the average.*

3.6 Biconnected Components

[JaJa, 78] provided an algorithm which uses the P-RAM model and finds the biconnected components of a graph in parallel time $O(\log n \cdot \log d \cdot \log k)$ where d is the depth of the graph and k is the number of biconnected components. The number of processors needed is $O(n^3/\log n)$. Again, this algorithm can be implemented on the RP-RAM model to run in $O(\log d \cdot \log k)$ parallel time with $O(n^3/\log n)$ processors. Our results of the Appendix about d and k imply then:

COROLLARY 3.2. *The expected parallel time complexity for finding biconnected components in the random graph* $G_{n,p}$ *with* $p \geq c/n$, $c > 5$, *is* $O((\text{loglog } n)^2)$ *for the RP-RAM model and* $O(\log n \cdot (\text{loglog } n)^2)$ *for the P-RAM. The number of processors needed is* $O(n^3/\log n)$. *The probability that the parallel time is more than this is* $O(2n^{-(c-1)})$.

4. EXPECTED PARALLEL TIME FOR MINIMUM WEIGHT SPANNING TREE

4.1 The Algorithm of Sollin

We consider here the problem of finding the minimum weight spanning tree of instances of the random graph $G_{n,p}$, $p \geq c/n$, the edges of which have positive costs drawn independently from a continuous distribution over a positive domain.

We apply here Sollin's algorithm (see [Papadimitriou, 77]). This algorithm was previously shown in [JaJa, 78] to have worst case parallel time $O(\log^3 n)$ on a P-RAM. The algorithm is based on the following observation: Let (V,T) be the minimum spanning tree (MST) of the graph $G = (V,E)$ under an edge cost mapping $c: E \rightarrow \mathbb{R}^+$ Then, for every $v \in V$, $e_v \in T$, where e_v is among the shortest of the edges incident to vertex $v \in V$.

Sollin's algorithm begins by finding all minimum-cost edges incident upon each node of G. These edges will constitute a forest if the cost function is one-one. This is so in our case because costs are drawn from a continuous distribution. We find the connected components of that forest and identify the vertices of each component. From here on, Sollin's algorithm applies the same process to the graph whose vertices are the components. The algorithm proceeds until one vertex remains.

4.2 Expected Time Analysis

Let us consider the SP-RAM model first. We can find the minimum edge incident to each vertex in time $O(n/p + \text{loglog } p)$ by using np processors. For $p = n/\text{loglog } n$ we get parallel time $O(\text{loglog } n)$ and need $n^2/\text{loglog } n$ processors for this stage. For any particular vertex pair $\{u,v\}$, the probability that this edge appears in the forst of the minimum edges is $p \cdot \text{prob}\{\{u,v\}$ has min cost of all edges out of u, given $\{u,v\}$ appeared$\}$ + $p \cdot \text{prob}\{\{u,v\}$ has minimum cost of all edges out of v, given $\{u,v\}$ appeared$\}$. Since weights are assigned independently, one can show that $\text{prob}\{\{u,v\}$ has minimum cost of all edges out of $u\} \geq 1/c$ for $p \geq c/n$. Hence, $\text{probability}\{\{u,v\}$ appears in the forest$\} \geq 2/n$. So, the constructed forest has embedded into it the instances of a random graph $G_{n,p'}$ with $p' \geq 2/n$. By the results of Appendix of this paper, the forest will have a big component of cardinality $>n-\log n$ and a small $O(\log n)$ number of smaller components, with probability $>1 - 1/n^2$.

The construction of the new graph (of the "nodes" being the components of the original with respect to minimum edges) will hence take $O(\text{loglog } n)$ time in the SP-RAM (and also RP-RAM) model, with high probability. To determine the edges of this reduced graph will take $O(\text{loglog } n)$ time for each vertex then $O(\text{loglog } n)$ time at most for all pairs of components. The connected components of the new graph can be found by a deterministic algorithm in $O(\text{loglog } n)$ time. This process will continue giving us an $O((\text{loglog } n)^2)$ time algorithm with probability $\geq 1 - n^{-2}$. Hence:

THEOREM 4.1. *The parallel expected time to find the MST of a random graph* $G_{n,p}$ *with random, independent weights in the SP-RAM (and so the RP-RAM) model, is* $O((\text{loglog } n)^2)$. *By simulation, the expected parallel time in the P-RAM is* $O(\log n \cdot (\text{loglog } n)^2)$. *The number of processors needed is* $O(n^2/\text{loglog } n)$. *The probability that the time is more than stated, is* $O(n^{-2})$.

5. EXPECTED PARALLEL TIME FOR ALL PAIRS SHORTEST PATHS

Given a weighted n-vertex graph G, the all pairs shortest-path matrix A is an $n \times n$ matrix such that $A(i,j)$ is the length of a shortest path from i to j in G. Let $A^k(i,j)$ denote the length of a shortest path from i to j going through at most k-intermediate vertices. Clearly $A(i,j) = A^d(i,j)$ where d = depth of G. Let $A^0(i,j)$ be the length of the edge $\{i,j\}$ if $\{i,j\}$ is in the graph, and $+\infty$ else. It is easy to see that $A^k(i,j) = \min_m \{A^{k/2}(i,m) + A^{k/2}(m,j)\}$. Hence we can compute A^d by computing A^2, $A^4, \ldots, A^{2^i} \ldots A^d$. A^k can be computed from $A^{k/2}$ using the matrix multiplication algorithm with + substituted for * and min for +. Since matrix multiplication can be done in time $O(\log n)$ in P-RAM using $n^3/\log n$ processors and in time $O(\log\log n)$ in SP-RAM (and so RP-RAM) using $n^3/\log n$ processors, we get an $O(\log n \log d)$ deterministic time for P-RAMs and an $O((\log\log n)\cdot\log d)$ for random graph $G_{n,p}$ with $p \geq c/n$ and $c > 1$ we get by our results on the depth of $G_{n,p}$: (See Theorem AI of the Appendix).

COROLLARY 5.1. *The expected parallel time for all pairs shortest paths in the* $G_{n,p}$ *model with* $p \geq c/n$, $c > 1$ *and arbitrary weights is* $O((\log\log n)^2)$ *for the SP-RAM or RP-RAM and* $O(\log n \cdot (\log\log n))$ *for the P-RAM model. The probability that the time is more than stated above is* $O(e^{-kn^{\alpha}})$, *for* $k, \alpha > 0$. *The number of processors needed is* $O(n^3/\log n)$.

Note [Dekel, Nassimi, Sahni, 81] gave an $O(\log^2 n)$ deterministic algorithm for the P-RAM model which uses n^3 processors.

6. EXPECTED PARALLEL TIME FOR RANDOM GRAPH ISOMORPHISM

The following algorithm, given by [Babai, Kucera, 79] is essentially a Breadth-First-Search procedure applied to random graphs of the model $G_{n,p}$ with $p = 1/2$:

[1] Classify vertex by valences (i.e., each vertex gets its valence as a label). Let $c_1,\ldots,c_h$ be the classes of the above classification, arranged by the order induced by this classification.

[2] (First refinement): Let $N_i(v)$ be the number of neighbors of v in c_i and let $N_i^4(v)$ = smallest nonnegative integer congruent to $N_i(v)$ mod 4. Then, two vertices v,u are ordered now if (a) they were of different label before or (b) they had same labels but

$$(N_1^4(v),\ldots,N_h^4(v)) < (N_1^4(u),\ldots,N_h^4(u))$$

lexicographically.

[3] Let $c_1,c_2,\ldots,c_h$, be the classes of [2]. Apply step [2] to these classes.

THEOREM 6.1. [Babai, Kucera, 79]. *Let* U *be the set of vertices whose class is not a singleton after step* [3]. *Then* $\text{Prob}\{|U| \geq 1\} \leq \exp(-cn)$, $c > 0$. *The valence classification can be done in* $O(\log\log n)$ *time by SP-RAM (and thus also the RP-RAM) by using a total of* $n^2/\log\log n$ *processors. In P-RAM, it can be done in time* $O(\log n)$ *by using* $O(n^2/\log n)$ *processors. The partial orders created by the refinements are best stored as a function, corresponding to them. Clearly a function computation is not more than* $O(\log\log n)$ *in SP-RAM (and thus also the RP-RAM) and* $O(\log n)$ *in P-RAM by using* $O(n^2/\log\log n)$ *and* $O(n^2/\log n)$ *processors, respectively. So:*

COROLLARY 6.1. *A canonical labelling algorithm on* $G_{n,p}$, $p = 1/2$, *takes average parallel time* $O(\log\log n)$ *in SP-RAM (and thus also the RP-RAM) with* $O(n^2/\log\log n)$ *processors and* $O(\log n)$ *time in P-RAM with* $O(n^2/\log n)$ *processors.*

7. EXPECTED SEQUENTIAL SPACE COMPLEXITY FOR GRAPH ALGORITHMS

THEOREM 7.1. *The upper bound on required sequential space for transitive closure on digraphs is* $O(\log n \log d)$ *where* d *is the depth of the digraph.*

Proof. The proof is similar to the techniques used by [Savitch, 70] for simulating nondeterministic space by deterministic space.

Input: digraph $D = (V,E)$, $|V| = n$.

The following procedure tests if there is a path from u to v, of length $\geq \ell \geq 0$.

```
Algorithm  CONNECT(u,v,ℓ)
begin
   if  (u,v) ∈ E  then return TRUE else
      begin
      for all  w ∈ V  if
         CONNECT(u,v,⌈ℓ/2⌉)  and  CONNECT(w,v,⌈ℓ/2⌉
         then return  TRUE
end
```

It is clear that we get the transitive closure by repeated applications of CONNECT. Each time we just remember the recursion depth. Therefore, the required sequential space is $O(\log n \log d)$ where d = depth of D. □

COROLLARY. *The average sequential space for transitive closure is* $O(\log n \log\log n)$. *The probability that is more than stated is* $O(\exp(-kn^{\alpha}))$.

Note that strong components, minimum cost paths etc. require only $O(\log n \log d)$ sequential space in the worst case, implying an average of $O(\log n \log\log n)$ for random graphs $G_{n,p}$ or digraphs $D_{n,p}$.

Note also that we can do isomorphism in $O(\log n)$ average sequential space. The reason is that we can find the rank of any out of n integers by using only $O(\log n)$ space.

REFERENCES

Angluin, D. and L.G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Paths and Matchings," *J. Comp. Syst. Sci.* 18 (1979), pp. 155-193.

L. Babai and L. Kucera, "Canonical Labelling of Graphs in Linear Average Time," CH1471-2/79, 1979, IEEE.

F. Chin, J. Lam, and I. Chen, "Optimal Parallel Algorithms for the Connected Components Problem," *Foundations of Computer Science* (FOCS), 1981.

E. Dekel, D. Nassimi, S. Sahni, "Parallel Matrix and Graph Algorithms," *SIAM J. Comp.* 10(4), Nov. 1981.

Erdös, P. and A. Renyi, "On the Evolution of Random Graphs," *Pub. Math. Inst. Hung Acad. Sci.* 5A, 1960, pp. 17-61.

D. Hirschberg, A. Chandra, D. Sarwate, "Computing Connected Components on Parallel Computers," *Commun. of the ACM* 22(8), August 1979.

J. Ja'Ja', "Graph Connectivity Problems on Parallel Computers," TR GS-78-05, Dept. of Computer Science, Penn. State Univ., PA, 1978.

R. M. Karp, "The Probabilistic Analysis of Combinatorial Search Algorithms," *Algorithms and Complexity: New Directions and Recent Results*, J.F. Traub, Ed. Acad. Press, New York, 1976, pp. 1-19.

R. M. Karp, and M. Sipser, "Maximum Matchings in Sparce Random Graphs," *Foundations of Computer Science*, 1981.

R. M. Karp and R. Tarjan, "Linear Expected Time Algorithms for Connectivity Problems," *Proc. of the 12th ACM Symp. on Theory of Computing*, Los Angeles, Calif., 1980, pp. 368-377.

C.H. Papadimitriou, "Unpublished Notes," Harvard University, 1977.

J. Reif, "Symmetric Complementation," *14th ACM Symposium on Theory of Computing* San Francisco, Calif., May 1982.

J. Reif and P. Spirakis, "Random Matroids," *Proc. of the 12th ACM Symp. on Theory of Computing*, Los Angeles, Calif., 1980, pp. 385-347, also rewritten as "Probabilistic Analysis of Random Extension-Rotation Algorithms," TR-28-81, Aiken Comp. Lab., Harvard University, 1981.

J. Reif, and P. Spirakis, "k-Connectivity in Random Undirected Graphs," TR-19-81, Aiken Comp. Lab., Harvard University, 1981.

C. Savage and J. Ja'Ja', "Fast, Efficient Parallel Algorithms for Some Graph Problems," *SIAM J. Comp.* 10(4), Nov. 1981.

W.J. Savitch, "Relationships between Nondeterministic and Deterministic Tape Complexities," *J. Comp. System Sciences* 4(2), 1970, pp. 177-192.

C.P. Schnorr, "An Algorithm for Transitive Closure with Linear Expected Time," *SIAM J. Comp.* 7, 1978, pp. 127-133.

Y. Shiloah, and V. Vishkin, "A O(log n) Parallel Connectivity Algorithms," to appear in *J. of Algorithms*.

P. Spirakis, "Probabilistic Algorithms, Algorithms with Random Inputs and Random Combinatorial Structures," Ph.D. Thesis, Harvard University, December 1981.

P. Spirakis, and J. Reif, "Strong k-Connectivity in Digraphs and Random Digraphs," TR-25-81, Aiken Comp. Lab., Harvard University, 1981.

J. Wyllie, "The Complexity of Parallel Computation," Ph.D. Thesis, Cornell University, 1979.

APPENDIX

RESULTS OF THE THEORY OF RANDOM GRAPHS

A.1 The Distribution of the Depth of a Random Graph

Let the depth $d(G)$ of a graph $G = (V,E)$ to be the $\max_{u,v \in V}\{d(u,v),2\}$ where $d(u,v)$ is the length of the shortest path between u and v, if they are connected ($-\infty$ otherwise). The depth of a random graph $G_{n,p}$ is the random variable $d(G_{n,p})$ whose values are the depths of the instances G of $G_{n,p}$. The above definition generalizes to digraphs in an obvious way. We now give an average case argument about d and then we discuss d's distribution.

LEMMA 1. *The average value of* $d(G_{n,p})$ *is* $O(\log n)$ *for* $G_{n,p}$ *with* $p \geq c/n$, *and* $c > 1$.

Proof. Let $G = (V,E)$ be an instance of $G_{n,p}$ and let u be a particular node of G. Assume we do Breadth-First-Search out of u in parallel, so that nodes at the same distance from u are reached at the same time t. Let B_t be # nodes at depth t, i.e. reached at time t. Let S_t be the average size at time t of the set of visited nodes. Clearly

$$S_{t+1} = S_t + B_{t+1}$$

and

$$B_{t+1} = pB_t(n - S_t).$$

From these two equations one can prove (by induction) that $B_k = \theta(p^k(n-1)^k)$ and $S_k = \Omega(1 + c + \ldots + c^k)$ for $p \geq c/n$, $c > 1$. Clearly, $S_{\bar{d}} \leq n$ for $\bar{d}$ = the average depth of $G_{n,p}$. Hence $\bar{d} = O(\log n)$.

LEMMA 2. *The probability that* $d(G_{n,p}) > \log n$ *is bounded above by* $O(e^{-kn^{\alpha}})$ *for* k,α *positive constants depending on* c, *for graphs* $G_{n,p}$ *with* $p \geq c/n$, $c > 1$. *This implies that* $\bar{d} = O(\log n)$.

Proof. Consider two specific vertices u,v of V. The probability of no path of length 1 between u,v is $\phi_1 = 1-p$. The probability of no path of length 2 is $\phi_2 = (1-p^2)^{n-2}$ since, for a path of length 2 to exist, at most $n-2$ possible nodes in $V-\{u,v\}$ can participate, each such path has probability p^2 to appear and these paths have no edges in common.

The probability of no path of length j is (for $j > 1$)

$$(1-p^j)(n-2)(n-3)\ldots(n-j)$$

since each such path has $\text{prob} = p^j$ to appear and there are $n-2$ independent choices for the first vertex after u, $n-3$ choices for the second vertex, etc. The probability that there is no path of length $\log n$ for the specific pair of vertices u,v is

$$\phi_{u,v} = \phi_1\,\phi_2 \cdots \phi_{\log n} = (1-p)(1-p^2)^{n-2}\ldots(1-p^{\log n})(n-2)\ldots(n-\log n)\,.$$

For $p \geq c/n$ and $p = o(1)$ we have $1/p \to \infty$ as $n \to \infty$ and hence

$$(1-p^k)(n-2)\ldots(n-k) \to e^{-p^k}(n-2)\ldots(n-k)\,.$$

For $p = o(1)$ we also have $1-p \to 1$ as $n \to \infty$. So,

$$\phi_{u,v} \to e^{-p[(n-2)p + \ldots + (n-2)\ldots(n-\log n)p^{\log n}]}$$

$$\leq e^{-p[(n-\log n)p + \ldots + (n-\log n)^{\log n}p^{\log n}]}$$

$$\leq e^{-p\left[\frac{[(n-\log n)p]^{\log n}-1}{(n-\log n)p-1} - 1\right]}$$

This expression is bounded above by $e^{-c[n^{c-1}/c - 1]}$ for $c > 1$. But

$$\text{Prob}\{d(G) > \log n\} \leqslant n^2 \phi_{u,v} \leqslant n^2 e^{-c[\frac{n^{c-1}}{c} - 1]} .$$

If $c > 1$ then $\exists k, \alpha$ depending on c $(k > 0, \alpha > 0)$ such that, for large n

$$n^2 e^{-n^{c-1}} \leqslant e^{-kn^{\alpha}} .$$

This proves the Lemma. □

Note that, for $pn \to \infty$ as $n \to \infty$, one can prove that the $d(G) \leqslant \frac{\log n}{\log(pn)}$ with probability tending to 1. For $p \geqslant c \frac{\log n}{n}$, there is almost always a path of length $\leqslant 3$ between any two nodes of $G_{n,p}$. (The reason is that, for any pair of vertices u,v of V, if S_u, (S_v) are the sets of vertices which are neighbors of u (of v) but not of v (not of u) then $|S_u| = |S_v| \geqslant \varepsilon \cdot n$ for some $\varepsilon > 0$ with probability $\geqslant 1 - e^{-\beta n}$, $\beta > 0$. Then, a lemma of [Erdös, Renyi, 59] (also [Karp, Tarjan, 80]) proves that at least one edge connects a node of S_u and a node of S_v with probability $\geqslant 1 - e^{-\beta' n}$ $0 < \beta' < \beta$.)

We will quote here some results about the valence of nodes of $G_{n,p}$, stated in [Erdös, Renyi, 59].

REMARK 1. If $p = c/n$ then the degree of any given vertex of an instance of $G_{n,p}$ has mean value $2c$ and the number of vertices having degree r is approximately

$$\frac{n(2c)^r e^{-2c}}{r!} .$$

REMARK 2. If $p(n) \geqslant \frac{\log n}{n} w(n)$ where $w(n) \to \infty$ as $n \to \infty$ then the probability that the degree of a vertex will be outside the interval

$2 \log n(1 \pm \varepsilon)w(n)$ is approximately $O(1/n^{\varepsilon^2 w(n)})$. Hence the probability that the degrees of not all n points will be between $2w(n)(1 \pm \varepsilon)\log n$, is tending to 0.

Note that, by an elementary application of Whitney's theorem, the connectivity k and depth d of a graph satisfy $k(d-1) \leqslant n$, if the graph is connected. This implies that with high (conditional on the fact that $G_{n,p}$ is connected) probability, the graph $G_{n,p}$ will be highly-connected for large values of p.

Note that all of the above results can be easily generalized for random digraphs $D_{n,p}$, $p \geqslant c/n$, $c > 2$.

A.2 The Size and Number of Connected Components for $p \geqslant c/n$

THEOREM A.2. *For any* $m = o(n)$ *there is a constant* $c_1 > 1$ *and a function* $t(n) > c_1 \log n/m$ *such that, if* $p > t(n)/n$ *then if* X *is the cardinality of the largest component of* $G_{n,p}$ *then*

$$\text{Prob}\{X \leqslant n - m\} \leqslant \frac{n}{e^{t(n)\cdot m}} \to 0 \qquad \text{as} \quad n \to \infty .$$

Proof. Assume that in the instance G of $G_{n,p}$ the cardinality X of the largest component satisfies the inequality $X \leqslant n - m$. Then, we can find two sets Y, Z such that $|Y| = m$, $|Z| = n - m$ and no edge between them. This event is above bounded by the probability $1-q$ where

q = Prob{for every pair of disjoint sets Y,Z of vertices of sizes m, $n-m$, there is at least one edge between Y,Z}.

Let us enumerate all possible pairs of sets of vertices of the above sizes. Call them

$$(Y_1,Z_1),(Y_2,Z_2),\ldots,(Y_g,Z_g)$$

where

$$g = \binom{n}{m}\binom{n-m}{n-m} = \binom{n}{m} \quad .$$

We have that

$$q = \text{Prob}\{E(Y_1,Z_1) \neq \phi \wedge \ldots \wedge E(Y_g,Z_g) \neq \phi\}$$

where $E(Y,Z)$ = set of edges between Y,Z. So, by Baye's formula,

$$\text{Prob}\{E(Y_1,Z_1)\neq\phi\} \cdot \text{Prob}\left\{\frac{E(Y_2,Z_2)\neq\phi}{E(Y_1,Z_1)\neq\phi}\right\} \ldots \text{Prob}\left\{\frac{E(Y_h,Z_g)\neq\phi}{\bigwedge_{i=1\ldots g-1} E(Y_i,Z_i)\neq\phi}\right\} \quad .$$

We need the following enumeration lemma:

LEMMA 3. *For every two sets* Y_i,Z_i *having at least one edge* e *between them, there are at least*

$$g_1 = \binom{n-2}{m-1}$$

pairs of sets of sizes m, n-m *which also have this edge between them.*

This lemma can be easily shown by taking out the two vertices of e and enumerating.

As a corollary, we conclude that there is a suitable enumeration of the sets in the q product such that for every term i not equal to 1 the next g_1 or more terms (conditioned on the existence of an edge from A_i to B_i) will be equal to 1. Hence, the value of q is

$$q \geq [\text{Prob}\{E(Y_1,Z_1) \neq \phi\}^{g/g_1}] \quad .$$

But

$$g/g_1 = \frac{n(n-1)}{m(n-m)} \leqslant \frac{n}{m} \quad .$$

Hence

$$q \geqslant [1 - (1-p)^{m(n-m)}]^{(n/m)}$$

$$\geqslant [1 - (1-p)^{1/p\ pm(n-m)}]^{(n/m)}$$

or

$$q \geqslant (1 - e^{-pm(n-m)})^{(n/m)} \geqslant 1 - \left(\frac{n}{m}\right) e^{-t(n)\cdot m}$$

or

$$q \geqslant 1 - e^{-[t(n)\cdot m - \log n]} > 1 - n^{-(c_1-1)} \quad .$$

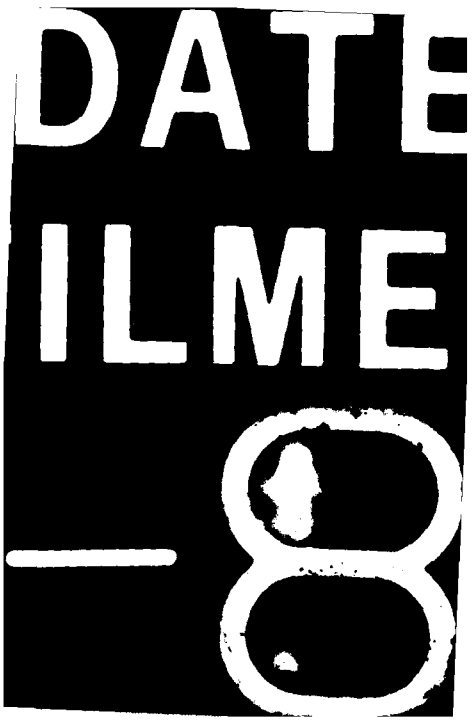
So, $q \to 1$ if $c_1 > 1$. So,

$$\text{Prob}\{X < n-m\} < e^{-[t(n)\cdot m - \log n]} \leqslant n^{-(c_1-1)} \to 0 \qquad \text{as} \quad n \to \infty \qquad \square$$

COROLLARY I. *For* $m = \log n$ *and* $t(n) \geqslant c_1 > 3$ *we get: The graph* $G_{n,p}$ *with* $p \geqslant c_1/n$ *has a component of size* $> n - \log n$ *with probability* $\geqslant 1 - n^{-(c_1-1)} > 1 - n^{-2}$.

COROLLARY II. *The graph* $G_{n,p}$ *with* $p \geqslant c/n$ *and* $c > 3$ *has less than* $\log n$ *connected components with probability* $\geqslant 1 - n^{-(c-1)} > 1 - n^{-2}$.

The above arguments can easily be generalized to *biconnected components* of a graph $G_{n,p}$ and *strong components* of a random digraph $D_{n,p}$ (and in fact to statements about size and number of k-blocks and k-strong blocks for any constant $k \geqslant 1$). See [Reif, Spirakis, 81] and [Spirakis, Reif, 81] for a proof of that.


DATE:
ILME
8